

#23

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants: Paul Freiburger et al.  
Assignee: Interval Research Corporation  
Title: Attention Manager for Occupying the Peripheral  
Attention of a Person in the Vicinity of a Display  
Device  
Serial No.: 08/620,641 Filed: March 22, 1996  
Examiner: Jeffery A. Brier Group Art Unit: 2775  
Attorney Docket No.: IR-003


-----  
Assistant Commissioner for Patents  
Washington, D. C. 20231

DECLARATION OF PHILIPPE P. PIERNOT  
UNDER 37 C.F.R. § 1.131

I, Philippe P. Piernot, hereby declare that:

1. I am an inventor of the invention of the above-referenced patent application.

2. Prior to October 19, 1995, I developed a computer program, an Applescript source code listing of which is attached hereto as Exhibit 1, that, together with the capabilities of conventional Internet browser software, acquired content data from a World Wide Web site and displayed an image generated from the content data as "wallpaper" on a display device of the computer ("content display computer") on which the computer program was executing. The browser software included a capability that allowed a user to select an image displayed at a Web site so as to cause the content data representing the image to be transferred from a data storage device of the Web site to the content display computer and stored at a user-designated

location of a non-volatile data storage device of the content display computer. In Exhibit 1, the user-designated location at which content data was stored is indicated at line 5. Line 6 caused execution of a set of instructions (see lines 23-34) that display an image or images generated from the content data. Line 29, together with lines 35-62, caused content data to be retrieved by the content display computer from an appropriate World Wide Web site. In particular, lines 39-41 identified multiple sets of content data to be retrieved (and displayed). Lines 50-54, together with lines 79-<sup>110</sup>~~120~~ <sup>PP</sup>, caused the sets of  content data to be successively retrieved and stored (see, in particular, line 87). Sets of content data were retrieved in alphabetical order of the name of the file containing the content data, in accordance with the manner in which an Applescript computer program orders a list of files within a folder defined on a data storage device (see line 37). Line 30, together with lines 63-78 and lines 134-161, caused identification of the format of a set of content data and display of the set of content data in accordance with the identified format. In the computer program shown in Exhibit 1, sets of content data in either the JPEG format (see lines 140-148) or the GIF format (see lines 150-159) could be used to generate an image display. Lines 31-33 caused the retrieved content data to be used to generate a display of the corresponding image or images: in particular, line 32 caused execution of a computer program called DeskPicture (a commercially available shareware computer program, produced by Peirce Software, that generated a display of an image as

"wallpaper" on a computer display screen) that accessed a set of content data from the appropriate (previously identified; see line 5, discussed above) location on the non-volatile data storage device and produced the corresponding image display. A set of content data was used to generate a display until a new set of content data was to be used to generate a display (the DeskPicture computer program included capabilities for displaying images generated from multiple sets of content data and specifying how long each set of content data was to be used to generate a display of an image), an updated version of the set of content data was to be used to generate a display, or operation of the computer program shown in Exhibit 1 terminated. Lines 10-22 caused the browser software to periodically retrieve (in Exhibit 1, every 5 minutes) and display an updated set of content data corresponding to a set of content data previously retrieved from a Web site. (An updated set of content data could be the same as the corresponding previously retrieved set of content data.)

3. Prior to October 19, 1995, I caused a computer-executable form of the computer program shown in Exhibit 1 to be stored on a first computer ("application management computer"). The application management computer was connected, using conventional hardware and software adapted for such purpose, to a second computer ("content display computer") such that instructions and/or data could be transferred from the application management computer to the content display computer. The presence of the computer-executable version of the computer

program on the application management computer was displayed on a display device of the content display computer. The content display computer was operated in accordance with conventional software that enabled a user of the content display computer to request transfer of the computer program from the application management computer to the content display computer and installation of the computer program on the content display computer. The content display computer was additionally connected, using conventional hardware and software adapted for such purpose, to the Internet computer network, such that the content display computer could be operated in accordance with conventional browser software to enable a user of the content display computer to select an image displayed at a Web site accessible via the Internet computer network so as to cause the content data representing the image to be transferred from a data storage device of the Web site to the content display computer and stored at a user-designated location of a non-volatile data storage device of the content display computer.

4. Prior to October 19, 1995, I caused a computer-executable form of a second computer program, similar to the computer program shown in Exhibit 1 (the "first computer program") and having capabilities similar to those described above in paragraph 2 of this Declaration, to be stored on the application management computer discussed above in paragraph 3 of this Declaration. The presence of the computer-executable version of the second computer program on the application management computer was displayed on a display device of the

content display computer. The content display computer discussed above in paragraph 3 of this Declaration was operated in accordance with conventional software that enabled a user of the content display computer to request transfer of the first or second computer program from the application management computer to the content display computer and <sup>effect</sup> ~~PR~~ installation of the first or ~~PR~~ second computer program on the content display computer. The second computer program differed from the first computer program in that the types of format of a set of content data that could be displayed were different from the types of format of a set of content data that could be displayed by the first computer program.

5. Prior to October 19, 1995, I developed a computer program, a MacroMedia Director source code listing of which is attached hereto as Exhibit 2, that, together with the capabilities of an Applescript program that I developed (described further below) and conventional Internet browser software, acquired content data from a World Wide Web site and displayed an image generated from the content data as a "screen saver" on a display device of the computer ("content display computer") on which the computer program was executing. The content display computer was operated in accordance with version 7 of the MacIntosh™ operating system. The browser software included a capability that allowed a user to select an image displayed at a Web site so as to cause the content data representing the image to be transferred from a data storage device of the Web site to the content display computer and stored

at a user-designated location of a non-volatile data storage device of the content display computer. In Exhibit 2, the user-designated location at which content data was stored is indicated at page 2, line 7. Lines 33-49 on page 6 of Exhibit 2 are a set of instructions that determined whether the screen saver was to be displayed or not. In particular, lines 38-43 prevented the screen saver from being displayed, while lines 45-46 caused the screen saver to be displayed if greater than a specified duration of time (which was user-specified in the computer program shown in Exhibit 2; see line 45 on page 6 of Exhibit 2 and control option 303 in Exhibit 3, discussed below) without interaction with the content display computer (an "idle period") had occurred. Lines 5-32 on page 2 of Exhibit 2 caused the display of one or more images generated from one or more sets of content data. More particularly, lines 5-12 on page 2 of Exhibit 2 determined which set of content data was to be used to generate image(s): each set of content data was used to generate images for a specified amount of time (which was user-specified in the computer program shown in Exhibit 2; see line 5 on page 2 of Exhibit 2 and control option 304 in Exhibit 3, discussed below). Lines 13-30 on page 2 of Exhibit 2 produced an image display from the set of content data identified in lines 5-12. Lines 33-38 on page 2 of Exhibit 2 caused, if appropriate, the screen saver to be turned off again. When the screen saver was turned off, the display shown in Exhibit 3 (discussed below) was produced on the display device of the content display computer using a display screen image definition file as defined using MacroMedia Director

constructs adapted for that purpose (see line 37 on page 2 of Exhibit 2). Lines 9-30 on page 6 of Exhibit 2 caused the computer program to periodically retrieve (in the computer program shown in Exhibit 2, within a daily ten minute window beginning at a user-specified time; see lines 10-17 on page 6 of Exhibit 2 and control option 305 in Exhibit 3, discussed below) a set of content data corresponding to Web site image(s) previously selected by a user (see lines 19-23 on page 6 of Exhibit 2). This periodic retrieval of content data occurred only when the screen saver was turned on (see lines 4-8 on page 6 of Exhibit 2, together with the above-mentioned lines 9-30 on page 6 of Exhibit 2). The actual retrieval of content data was accomplished at line 23 using an Applescript computer program called "fetchImages" (which is not shown as part of Exhibit 2) that accessed the user-designated location(s) of the non-volatile data storage device of the content display computer at which content data was stored to identify the World Wide Web site(s) (identification(s), e.g., URL(s), of which were stored together with the corresponding content data) from which the content data was obtained, then caused the browser software to retrieve content data from those site(s). I developed "fetchImages," which embodied the functionality of lines 29, 30, 35-62, 63-78, 79-120 and 134-161 of the computer program shown in Exhibit 1, to enable the Macromedia Director computer program shown in Exhibit 2 to make use of the browser software to transfer set(s) of content data from Web site(s) to the content display computer. (The Macromedia Director computer program

shown in Exhibit 2 could not communicate directly with the browser software, but could communicate with an Applescript computer program.)

6. Exhibit 3 depicts a display produced on the display device of the content display computer referred to above in paragraph 5 by the computer program shown in Exhibit 2 (see line 37 on page 2 of Exhibit 2, discussed above) when the screen saver was turned off. The display provided a graphical mechanism for enabling a user of the content display computer to control aspects of the operation of the computer program shown in Exhibit 2. A dialog box (designated by the numeral 301 in Exhibit 3) within the display included four control options that each enabled control of a corresponding aspect of the operation of the computer program shown in Exhibit 2. The first control option (designated by the numeral 302 in Exhibit 3) enabled the user to specify whether the screen saver would be displayed after detection of an idle period. The second control option (designated by the numeral 303 in Exhibit 3) enabled the user to specify the duration of time without interaction with the content display computer which had to pass before the screen saver would be displayed. The third control option (designated by the numeral 304 in Exhibit 3) enabled the user to specify the duration of time for which each set of content data would be used to generate an image display during operation of the screen saver. The fourth control option (designated by the numeral 305 in Exhibit 3) enabled the user to specify the time at which to begin retrieval each day of set(s) of content data corresponding



to Web site image(s) previously selected by a user.

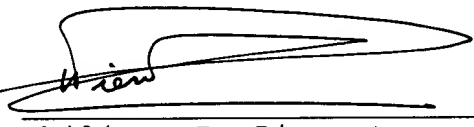
7. Prior to October 19, 1995, I developed a computer program, an Applescript source code listing of which is attached hereto as Exhibit 4, that, together with the capabilities of conventional Internet browser software, acquired content data from a World Wide Web site and displayed an image generated from the content data on a display device of the computer ("content display computer") on which the computer program was executing. The browser software included a capability that allowed a user to select an image displayed at a Web site so as to cause the content data representing the image to be transferred from a data storage device of the Web site to the content display computer. In Exhibit 4, line 4 caused execution of a set of instructions (see lines 21-28) that, in turn, caused the execution of still other sets of instructions to display an image or images generated from the content data. Depending on the type of content data acquired, the image was displayed as "wallpaper" (see line 25 and lines 29-49) or in a display area dedicated to the browser software (see line 26 and lines 50-64). In the former case (i.e., lines 25 and 29-49), lines 44 and 67-89 caused content data to be retrieved by the content display computer for use in generating an image display. After acquisition of the content data, the content data was stored at a user-designated location of a non-volatile data storage device of the content display computer. Lines 46-48 caused the retrieved content data to be used to generate a display of the corresponding image or images: in particular, line 47 caused execution of the computer

program called DeskPicture, as described above in paragraph 2, that produced the image display. In the latter case (i.e., lines 26 and 50-64), the computer program shown in Exhibit 4 did not cause content data to be stored on the non-volatile data storage device of the content display computer, but only used the content data to generate an image display immediately upon acquisition.

8. The acts described above in numbered paragraphs 2 through 7 were carried out in the United States.

9. I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Date: 6/4, 1999

  
Philippe P. Piernot

# Exhibit 1

```
1  property justLoaded : false
2  property folderPath : ""
3  property triggerMin : 0

4  on run
5      set folderPath to ((path to (the preferences folder)) as string) & "WebPictures:"
6      doIt()
7      set triggerMin to ((time of (current date)) / minutes) + 5
8      set justLoaded to true
9  end run

10 on idle
11     set mins to (time of (current date)) / minutes
12     if mins > triggerMin + 5 then
13         if justLoaded then
14             set justLoaded to false
15         end if
16     else
17         if not justLoaded and mins ≥ triggerMin then
18             doIt()
19             set triggerMin to ((time of (current date)) / minutes) + 5
20             --set justLoaded to true
21         end if
22     end if
23 end idle

23 on doIt()
24     set wasDeskPictureRunning to isProcessRunning("CLY7")
25     if wasDeskPictureRunning then
26         tell application "DeskPicture"
27             to quit
28         end if
29     else
30         set fileList to (list folder folderPath)
31         fetchAllPicturesIn(folderPath)
32         convertToPictAllPicturesIn(folderPath, fileList)
33     end if
34 end doIt

35 on fetchAllPicturesIn(folderPath)
36     set wasFrontierRunning to isProcessRunning("LAND")
37     set fileList to (list folder of folderPath)
38     set urlList to {}
39     repeat with fileName in fileList
40         set urlList to urlList & getFileComment(alias (folderPath & fileName))
```

# Exhibit 1

```

41 end repeat
42 if not wasFrontierRunning then
43     tell application "Frontier"
44         to quit
45     end if
46
47 set wasNetscapeRunning to isProcessRunning("MOSS")
48 -- Asks Netscape not to display alert boxes
49 tell application "Netscape Navigator™ 3.0"
50     set netscapeAlertApp to the alert application
51     set alert application to "zzzz"
52 end tell
53
54 repeat with i from 1 to (length of fileList)
55     set fileName to item i of fileList
56     set myURL to item i of urlList
57     netscapeGetURL(myURL, (folderPath & fileName & "1"), 5, 300)
58 end repeat
59
60 if wasNetscapeRunning then
61     -- Resume Netscape alert boxes display handling
62     tell application "Netscape Navigator™ 3.0"
63         set alert application to netscapeAlertApp
64     end tell
65 else
66     tell application "Netscape Navigator™ 3.0" to quit
67 end if
68 end fetchAllPicturesIn
69
70 on convertToPictAllPicturesIn(folderPath, fileList)
71     set wasClip2GifRunning to isProcessRunning("c2gf")
72     set wasJPegViewRunning to isProcessRunning("JVWR")
73
74     repeat with fileName in fileList
75         set fileAlias to (alias (folderPath & fileName))
76         convertToPict(folderPath & fileName & "1", fileName & "1")
77         «event ScTIExch» (alias (folderPath & fileName & "1")) given «class with»:(fileAlias)
78         «event ScTIdel» (alias (folderPath & fileName & "1"))
79     end repeat
80
81     if not wasClip2GifRunning then
82         tell application "Clip 2 Gif"
83             to quit
84         end if
85     if not wasJPegViewRunning then
86         tell application "JPegView"
87             to quit
88         end if
89     end
90 end convertToPictAllPicturesIn

```

## -----NETSCAPE RELATED ROUTINES-----

```

79 on netscapeGetURL(myLoc, destFile, nbOfTries, myTimeOut)
80     set errCounter to 0
81     repeat while errCounter < nbOfTries

```

82 tell application "Netscape Navigator™ 3.0"  
83 with timeout of myTimeOut seconds

Exhibit 1

84 repeat while the busy of window 1 ≠ 0

85 end repeat

86 set isLoaded to true

87 GetURL myLoc to (file destFile)

88 set isLoaded to false

89 repeat while not isLoaded

90 try

91 the busy of window 1

92 set isLoaded to true

93 on error

94 end try

95 end repeat

96 end timeout

97 try

98 if the file type of (info for (file destFile)) = "TEXT" then

99 set errCounter to errCounter + 1

100 «event ScTidele» destFile

101 else

102 return false -- no error

103 end if

104 on error

105 set errCounter to errCounter + 1

106 end try

107 end tell

108 end repeat

109 return true -- error

110 end netscapeGetURL

111 on «event WWW?PRBG»

112 return 1

113 end «event WWW?PRBG»

114 on «event WWW?PRMK»

115 return 0

116 end «event WWW?PRMK»

117 on «event WWW?PREN»

118 set finished to true

119 return 0

120 end «event WWW?PREN»

-----FINDER RELATED ROUTINES-----

121 on isProcessRunning(procString)

122 repeat with processName in (list processes)

123 if signature of (get process processName) = procString then

124 return true

125 end if

126 end repeat

127 return false

128 end isProcessRunning

```

129 on getFileComment(fileAlias)
130   tell application "Frontier"
131     ifile.getComment1(fileAlias)
132   end tell
133 end getFileComment

```

-----PICTURE CONVERSION ROUTINE-----

```

134 on convertToPict(filePath, fileName)
135   try --We check whether the file exists
136     set fileType to the file type of (info for (file filePath))
137   on error
138     return
139   end try
140   if fileType = "JPEG" then
141     tell application "Jpeg View"
142       try
143         open (alias filePath)
144         save document 1 in (alias filePath) as picture
145         close document 1
146       on error
147       end try
148     end tell
149   else
150     if fileType = "GIF" then
151       tell application "Clip 2 Gif"
152         try
153           open (file filePath) given «class fltp»:picture, «class kfil»:(file (filePath & "2"))
154           «event ScTIdete» (alias filePath)
155           «event ScTIRena» (alias (filePath & "2")) given «class name»:fileName
156         on error
157         end try
158       end tell
159     end if
160   end if
161 end convertToPict

```

## Exhibit 2

```
on exitFrame
  global gRunning, gLastScreenUpdate

  if desiredScreenSaverState() then
    set gLastScreenUpdate to 0
    initRearWindow()
    savePreferences()
    installMenu          -- removes the menubar
    convertPicturesIfNeeded()
    activate()
    set gRunning to TRUE
    go to frame "SlideShow"
  else
    go to the frame
  end if
end exitFrame
```

## Exhibit 2

```
1  on exitFrame
2    global gScreenNumber, gScreenCastNum, gRunning, gLastActivity, gFolderPath,
3    gLastScreenUpdate
4    if desiredScreenSaverState() then
5      if (the ticks - gLastScreenUpdate > 60 * value(the text of cast "DisplayTime")) then
6        set gScreenNumber to gScreenNumber + 1
7        set folderPath to gFolderPath & "Screen Saver Files:"
8        set fileName to getNthFileNameInFolder(folderPath, gScreenNumber)
9        if fileName = EMPTY then
10          set gScreenNumber to 1
11          set fileName to getNthFileNameInFolder(folderPath, gScreenNumber)
12        end if
13      if fileName <> EMPTY then
14        if (getFileType(folderPath & fileName) starts "PICT") then
15          if the castNum of sprite 2 = 5 then
16            set gScreenCastNum to 6
17          else
18            set gScreenCastNum to 5
19          end if
20          puppetSprite 2, TRUE
21          set the fileName of cast gScreenCastNum to folderPath & fileName
22          set pict to the picture of cast gScreenCastNum -- so that the castRect is
updated
23          set pict to 0 -- just in case :-)
24          set the castNum of sprite 2 to gScreenCastNum
25          set the locH of sprite 2 to (the stageRight - the stageLeft - the width of cast
gScreenCastNum) / 2
26          set the locV of sprite 2 to (the stageBottom - the stageTop - the height of
cast gScreenCastNum) / 2
27          puppetTransition random(49), 4, 10, FALSE
28          set gLastScreenUpdate to the ticks
29        end if
30      end if
31    end if
32    go to the frame
33  else
34    set gRunning to FALSE
35    releaseRearWindow()
36    installMenu cast "Menubar"
37    go to frame "UI"
38  end if
39 end exitFrame
```



## Exhibit 2

-----UTILITY FUNCTIONS-----

```
on filesIn folderPath
  put [] into fileList
  repeat with i = 1 to the maxInteger
    set fileName to getNthFileNameInFolder(folderPath, i)
    if fileName = EMPTY then exit repeat
    append(fileList, fileName)
  end repeat
  return fileList
end filesIn
```

```
on deleteFile filePath
  set fileIOXObj to FileIO(mNew, "read", filePath)
  return fileIOXObj(mDelete)
end deleteFile
```

```
on deleteContentOfFolder folderPath
  set fileList to filesIn(folderPath)
  repeat with fileName in fileList
    deleteFile(folderPath & fileName)
  end repeat
end deleteContentOfFolder
```

```
on newUniqueFileNameIn folderPath
  set counter to -1
  set done to false
  set fileList to filesIn(folderPath)
  repeat while not done
    set counter to counter + 1
    if not getOne(fileList, "" & counter) then
      set done to true
    end if
  end repeat
  return "" & counter
end newUniqueFileNameIn
```

```
on replaceFilesKeepingComments srcFolderPath, dstFolderPath
  set srcFileList to filesIn(srcFolderPath)
  set dstFileList to filesIn(dstFolderPath)
  repeat with fileName in srcFileList
    if getOne(dstFileList, fileName) then
      set comment to getFileComment(dstFolderPath & fileName)
      deleteFile(dstFolderPath & fileName)
      moveFile(srcFolderPath & fileName, dstFolderPath)
      setFileComment(dstFolderPath & fileName, comment)
    else
      moveFile(srcFolderPath & fileName, dstFolderPath)
    end if
  end repeat
end moveFiles
```

## Exhibit 2

---

```
on getFileComment filePath
  set comment to GetComment(filePath)
  set zeroChar to numToChar(0)
  set theLength to the length of comment
  set done to false
  set i to 1
  repeat while not done
    if (i = theLength) or ((char i of comment) = zeroChar) then
      set done to true
    else
      set i to i + 1
    end if
  end repeat
  if i <= 1 then
    return ""
  else
    return char 1 to i - 1 of comment
  end if
end getFileComment
```

---

```
on setFileComment filePath, name
  SetComment(filePath, name)
end setFileComment
```

---

```
on renameFile filePath, newName
  set oldDelim to the itemDelimiter
  set the itemDelimiter to ":"
  set fileName to the last item of filePath
  set the itemDelimiter to oldDelim
  set folderPathEnd to (the length of filePath) - (the length of fileName)
  set foldPath to (char 1 to folderPathEnd of filePath)

  FSRename(filePath, foldPath & newName)
end renameFile
```

---

```
on moveFile filePath, dstFolderPath
  FSCatMove(filePath, dstFolderPath)
end moveFile
```

---

```
on getFileType filePath
  set fileIOXObj to FileIO(mNew, "read", filePath)
  set type to fileIOXObj(mGetFinderInfo)
  fileIOXObj(mDispose)
  return type
end getFileType
```

---

## Exhibit 2

```
on isProcessRunning procString
  thePrograms "", procString
  return charToNum(char 1 of the result) <> 0
end isProcessRunning
```

---

```
on activate
  open the moviePath & the movieName
end activate
```

---

```
on getSecondsSinceMidnight
  global gTimeObj

  return gTimeObj(mGetSecsSinceMidnight)
end getSecondsSinceMidnight
```

## Exhibit 2

```
1 on idle
2   global gRunning, gMode, gFolderPath, gFetched
3   if gRunning then
4     if not(desiredScreenSaverState()) then
5       set gRunning to FALSE
6       releaseRearWindow()
7       installMenu cast "Menubar"
8       go to frame "UI"
9     else
10      set hours to value(the text of cast "hours")
11      if the text of cast "am/pm" = "PM" then
12        if hours < 12 then
13          set hours to hours + 12
14        end if
15      end if
16      set downloadTime to (3600 * hours) + (60 * value(the text of cast "minutes"))
17      if gFetched = 0 and gMode = "Done" and getSecondsSinceMidnight() > downloadTime and
      getSecondsSinceMidnight() < (downloadTime + 600) then
18        set gFetched to the ticks
19        deleteContentOfFolder(gFolderPath & "Temporary Files:downloaded:")
20        deleteContentOfFolder(gFolderPath & "Temporary Files:temp:")
21        deleteContentOfFolder(gFolderPath & "Temporary Files:converted:")
22        set gMode to "FetchAndConvert"
23        open the moviePath & "Helper Apps:fetchImages"
24      else
25        if gFetched <> 0 and the ticks - gFetched > 36000 then -- we should be done
26          downloading
27            set gFetched to 0
28          end if
29        end if
30      end if
31    pass
32  end idle

33 on desiredScreenSaverState
34   global gLastActivity, gLastMouseH, gLastMouseV, gLastKeyCode, gKeyDetectorXObj
35   set mH to the mouseH
36   set mV to the mouseV
37   set kc to the keyCode
38   if not (the hilite of cast "on/off") or the mouseDown or mH <> gLastMouseH or mV <>
39   gLastMouseV or gLastKeyCode <> kc or gKeyDetectorXObj(mCheckKey) <> 0 then
40     set gLastMouseH to mH
41     set gLastMouseV to mV
42     set gLastKeyCode to kc
43     set gLastActivity to the ticks
44     return FALSE
45   else
46     if the ticks - gLastActivity > 3600 * value(the text of cast "SleepDelay") then
47       return TRUE
48     end if
49   end if
50 end desiredScreenSaverState
```

## Exhibit 2

```
50 on startMovie
51   global gMode, gTimeObj, gKeyDetectorXObj, gMiscXObj, gLastScreenUpdate,
52   gScreenDisplayTime, -
53   gScreenNumber, gRunning, gFolderPath, gFetched

54   set gScreenNumber to 0
55   set gFetched to 0
56   set gRunning to FALSE
57   set gMode to "Done"
58   set the hilite of cast "on/off" to TRUE
59   set gLastScreenUpdate to 0
60   set gScreenDisplayTime to 600

61   set gTimeObj to TimeSinceMidnight( mNew )
62   set gKeyDetectorXObj to KeyDetector(mNew)
63   set gMiscXObj to misc_x(mNew)

64   set gFolderPath to gMiscXObj(mPrefsFolder) & "NetScreen:"
65   installMenu cast "Menubar"
66   loadPreferences()

67   --put callbackFactory(mNew) into callbackObject
68   --setCallback RunOSAScript, callbackObject
69   --RunOSAScript("open")
70 end startMovie

71 on stopMovie
72   global gTimeObj, gKeyDetectorXObj, gMiscXObj

73   savePreferences()
74   if objectP(gTimeObj) then
75     gTimeObj(mDispose)
76   end if
77   if objectP(gKeyDetectorXObj) then
78     gKeyDetectorXObj(mDispose)
79   end if
80   if objectP(gMiscXObj) then
81     gMiscXObj(mDispose)
82   end if
83   releaseRearWindow()

84   --RunOSAScript("close")
85   --callBackFactory(mDispose)
86 end stopMovie

87 on convertPicturesIfNeeded
88   global gMode, gFolderPath

89   if gMode = "Done" then
90     set files to filesToConvert()
91     if files <> EMPTY then
92       deleteContentOfFolder(gFolderPath & "Temporary Files:downloaded:")
93       deleteContentOfFolder(gFolderPath & "Temporary Files:temp:")
94       deleteContentOfFolder(gFolderPath & "Temporary Files:converted:")
95       repeat with fileName in files
96         moveFile(gFolderPath & "Screen Saver Files:" & fileName, -
97   gFolderPath & "Temporary Files:downloaded:")
```

## Exhibit 2

```
98         end repeat
99         set gMode to "Convert"
100        open the moviePath & "Helper Apps:fetchImages"
101        end if
102        end if
103    end convertPicturesIfNeeded

104    on filesToConvert
105        global gFolderPath

106        set folderPath to gFolderPath & "Screen Saver Files:"
107        set fileList to filesIn(folderPath)
108        set files to []
109        repeat with fileName in fileList
110            set type to getFileType(folderPath & fileName)
111            if not (type starts "PICT") then
112                append files, fileName
113            end if
114        end repeat
115        return files
116    end filesToConvert

117    on quitNetScreen
118        stopMovie()
119        quit
120    end quitNetScreen

-----

121    on getStatus
122        global gFolderPath, gMode

123        if voidP(gMode) then
124            set gMode to "Done"
125        end if
126        set folderPath to gFolderPath & "Screen Saver Files:"
127        set status to gMode & " " & ¬
128        isProcessRunning("MOSS") & " " & ¬
129        isProcessRunning("c2gf")
130        if gMode = "FetchAndConvert" then
131            set fileList to filesIn(folderPath)
132            repeat with fileName in fileList
133                set status to status & RETURN & fileName & RETURN & ¬
134                getFileComment(folderPath & fileName)
135            end repeat
136        end if
137        return status
138    end getStatus

-----

139    on ScriptDone
140        global gFolderPath, gMode

141        if gMode = "FetchAndConvert" then
142            replaceFilesKeepingComments(gFolderPath & "Temporary Files:converted:", ¬
143            gFolderPath & "Screen Saver Files:")
144        else
145            if gMode = "Convert" then
```

## Exhibit 2

```
145      set files to filesIn(gFolderPath & "Temporary Files:downloaded:")
146      repeat with fileName in files
147          set comment to getFileComment(gFolderPath & "Temporary Files:downloaded:" &
fileName)
148          setFileComment(gFolderPath & "Temporary Files:converted:" & fileName, comment)
149      end repeat
150      replaceFilesKeepingComments(gFolderPath & "Temporary Files:converted:", -
gFolderPath & "Screen Saver Files:")
151      end if
152      end if
153      deleteContentOfFolder(gFolderPath & "Temporary Files:downloaded:")
154      deleteContentOfFolder(gFolderPath & "Temporary Files:temp:")
155      deleteContentOfFolder(gFolderPath & "Temporary Files:converted:")
156      set gMode to "Done"
157      activate()
158  end ScriptDone

159  on loadPreferences
160      global gFolderPath

161      set prefPath to gFolderPath & "NetScreen.prefs"
162      set fileXObj to FileIO(mNew, "read", prefPath)
163      set l to fileXObj(mReadLine)
164      set the hilite of cast "on/off" to value(word 2 of l)
165      set l to fileXObj(mReadLine)
166      set the text of cast "SleepDelay" to word 2 of l
167      set l to fileXObj(mReadLine)
168      set the text of cast "DisplayTime" to word 2 of l
169      set l to fileXObj(mReadLine)
170      set the text of cast "hours" to word 2 of l
171      set the text of cast "minutes" to word 3 of l
172      set the text of cast "am/pm" to word 4 of l
173      fileXObj(mDispose)
174  end loadPreferences

175  on savePreferences
176      global gFolderPath

177      set prefPath to gFolderPath & "NetScreen.prefs"
178      set fileXObj to FileIO(mNew, "write", prefPath)
179      fileXObj(mWriteString, "on/off " & the hilite of cast "on/off" & RETURN)
180      fileXObj(mWriteString, "SleepDelay " & the text of cast "SleepDelay" & RETURN)
181      fileXObj(mWriteString, "DisplayTime " & the text of cast "DisplayTime" & RETURN)
182      fileXObj(mWriteString, "DownloadTime " & the text of cast "hours" & " " & -
the text of cast "minutes" & " " & the text of cast "am/pm" & RETURN)
183      fileXObj(mDispose)
184  end savePreferences

185  -- Factory: MISC_X ID:10001
186  -- Misc_X, Misc Utils XObject, v1.1.3
187  --I      mNew
188  --S      mBootName
189  --S      mSystemFolder
190  --S      mPrefsFolder
191  --IS     mFileExists, fp
192  --ISS    mCopyFile, sP, dP
193  --IS     mFolderExists, fp
194  --IS     mInsureFolder, fp
```

195 --XS mDeleteFolder, fP.  
196 --SS mFolderList, fP  
197 --SSSSS mAsk, q, dR, bOk, bCan  
198 --SSSSS mAnswer, q, bL, bM, bR  
199 --IS mSpaceOnVol, vN  
200 --X. mFlushActions



```

global gRwObj

on initRearWindow
  if objectP(gRwObj) then
    gRwObj(mDispose)
  end if

  if createRwObject() >= 0 then
    gRwObj(mPatToWindow, -5)
  end if
end initRearWindow

on releaseRearWindow
  if objectP(gRwObj) then
    gRwObj(mDispose)
  end if
end releaseRearWindow

on createRwObject
  if not objectP(gRwObj) then
    -- "M" indicates multiple monitors, "S" is for single monitor configuration.
    -- ONLY use "S" if there is not enough room for multiple monitors.
    -- So first...let's try it with multiple-monitor configuration:
    set gRwObj = RearWindow(mNew, "M")
    set error to value(gRwObj)
    if error < 0 then
      gRwObj(mDispose)
      return error
    end if
    if the freeBlock < gRwObj(mGetMemoryNeeded) then
      -- delete the object and create it again with a single-monitor config...
      if objectP(gRwObj) then
        gRwObj(mDispose)
        set gRwObj = RearWindow(mNew, "S")
      end if
      set error to value(gRwObj)
      if error < 0 then
        gRwObj(mDispose)
        return error
      end if
    end if
  end if
  return value(gRwObj)
end createRwObject

```

-- Paint in back

## Exhibit 2

```
global gRwObj

on initRearWindow
  if objectP(gRwObj) then
    gRwObj(mDispose)
  end if

  if createRwObject() >= 0 then
    gRwObj(mPatToWindow, -5)
  end if
end initRearWindow

on releaseRearWindow
  if objectP(gRwObj) then
    gRwObj(mDispose)
  end if
end releaseRearWindow

on createRwObject
  if not objectP(gRwObj) then
    -- "M" indicates multiple monitors, "S" is for single monitor configuration.
    -- ONLY use "S" if there is not enough room for multiple monitors.
    -- So first...let's try it with multiple-monitor configuration:
    set gRwObj = RearWindow(mNew, "M")
    set error to value(gRwObj)
    if error < 0 then
      gRwObj(mDispose)
      return error
    end if
    if the freeBlock < gRwObj(mGetMemoryNeeded) then
      -- delete the object and create it again with a single-monitor config...
      if objectP(gRwObj) then
        gRwObj(mDispose)
        set gRwObj = RearWindow(mNew, "S")
      end if
      set error to value(gRwObj)
      if error < 0 then
        gRwObj(mDispose)
        return error
      end if
    end if
  end if
  return value(gRwObj)
end createRwObject
```

```
--factory callBackFactory
--method mNew
--  me(mPut, 1, "SendCardMessage")
--  me(mPut, 2, "EvalExpr")
--  me(mPut, 3, "StringLength")
--  me(mPut, 4, "StringMatch")
--  me(mPut, 5, "SendHCMessage")
--  me(mPut, 6, "ZeroBytes")
--  me(mPut, 7, "PasToZero")
--  me(mPut, 8, "ZeroToPas")
--  me(mPut, 9, "StrToLong")
--  me(mPut, 10, "StrToNum")
--  me(mPut, 11, "StrToBool")
--  me(mPut, 12, "StrToExt")
--  me(mPut, 13, "LongToStr")
--  me(mPut, 14, "NumToStr")
--  me(mPut, 15, "NumToHex")
--  me(mPut, 16, "BoolToStr")
--  me(mPut, 17, "ExtToStr")
--  me(mPut, 18, "GetGlobal")
--  me(mPut, 19, "SetGlobal")
--  me(mPut, 20, "GetFieldByName")
--  me(mPut, 21, "GetFieldByNum")
--  me(mPut, 22, "GetFieldByID")
--  me(mPut, 23, "SetFieldByName")
--  me(mPut, 24, "SetFieldByNum")
--  me(mPut, 25, "SetFieldById")
--  me(mPut, 26, "StringEqual")
--  me(mPut, 27, "ReturnToPas")
--  me(mPut, 28, "ScanToReturn")
--  me(mPut, 31, "FormatScript")
--  me(mPut, 32, "ZeroTermHandle")
--  me(mPut, 33, "PrintTEHandle")
--  me(mPut, 34, "SendHCEvent")
--  me(mPut, 35, "HCWordBreakProc")
--  me(mPut, 36, "BeginXSound")
--  me(mPut, 37, "EndXSound")
--  me(mPut, 38, "RunHandler")
--  me(mPut, 39, "ScanToZero")
--  me(mPut, 40, "GetXResInfo")
--  me(mPut, 41, "GetFilePath")
--  me(mPut, 42, "FrontDocWindow")
--  me(mPut, 43, "PointToStr")
--  me(mPut, 44, "RectToStr")
--  me(mPut, 45, "StrToPoint")
--  me(mPut, 46, "StrToPoint")
--  me(mPut, 47, "GetFieldTE")
--  me(mPut, 48, "SetFieldTE")
--  me(mPut, 49, "GetObjectName")
--  me(mPut, 50, "GetObjectScript")
--  me(mPut, 51, "SetObjectScript")
--  me(mPut, 52, "StackNameToNum")
--  me(mPut, 53, "Notify")
--  me(mPut, 54, "SowHCAAlert")
--  me(mPut, 100, "NewXWindow/GetNewXWindow")
--  me(mPut, 101, "CloseXWindow")
--  me(mPut, 102, "SetXWIdleTime")
--  me(mPut, 103, "XWHasInterruptCode")
--  me(mPut, 104, "RegisterXWMenu")
```

# Exhibit 2

```
-- me(mPut, 105, "BeginXWEdit/EndXWedit")
-- me(mPut, 106, "SaveXWScript")
-- me(mPut, 107, "GetCheckPoints")
-- me(mPut, 108, "SetCheckPoint")
-- me(mPut, 109, "XWAllowReEntrancy")
-- me(mPut, 110, "SendWindowMessage")
-- me(mPut, 111, "HideHCPalettes")
-- me(mPut, 112, "ShowHCPalettes")
-- me(mPut, 113, "XWAlwaysMoveHigh")
-- me(mPut, 200, "GoScript")
-- me(mPut, 201, "StepScript")
-- me(mPut, 202, "AbortScript")
-- me(mPut, 203, "CountHandlerInfo")
-- me(mPut, 204, "GetHandlerInfo")
-- me(mPut, 205, "GetVarInfo")
-- me(mPut, 206, "SetVarValue")
-- me(mPut, 207, "GetStackCrawl")
-- me(mPut, 208, "TraceScript")
--
--method mEvalExpr x
-- -- put "mEvalExpr" && x
-- -- if x = "cd fld " & QUOTE & "urlField" & QUOTE then
-- --     return "tell application " & QUOTE & "Netscape" & QUOTE & " to make new window"
-- -- else
-- --     if x = "the name of cd fld " & QUOTE & "urlField" & QUOTE then
-- --         put "beep"
-- --         return "urlField"
-- --     else
-- --         if x = "the id of cd fld " & QUOTE & "urlField" & QUOTE then
-- --             put "beep beep"
-- --             --return 100
-- --         end if
-- --     end if
-- -- end if
-- if word 1 of x = "----" then
--     return "tell me to activate"
-- end if
--end mEvalExpr
--
--method mEvalExpr x
-- put "mEvalExpr" && x
-- if the length of x >= 10 then
--     set s to char 1 to 10 of x
--     if (s <> "the id of ") and (s <> "the name o") then
--         return x
--     end if
-- end if
--end mEvalExpr
--
--method mSendHCMMessage x
--put "mSendHCMMessage" && x
--
--method mSendCardMessage x
--put "mSendCardMessage" && x
--
--method mGetFieldByName card, name
--put "mGetFieldByName" && card && name
--
--method mGetFieldByNum card, Num
```

## Exhibit 2

```
--put "mGetFieldByNum" && card && num
--
--method mGetFieldByID card, id
--put "mGetFieldByID" && card && id
--
--method mSetFieldByName card, name, value
--put "mSetFieldByName" && card && name && value
--
--method mSetFieldByNum card, num, value
--put "mSetFieldByNum" && card && num && value
--
--method mSetFieldByID card, id, value
--put "mSetFieldByID" &7 card && id && value
--
--method mGetFieldTE
--put "mGetFieldTE" --&& arg1 && arg2 && arg3
--
--method mUnknown which
--put me(mGet, value(which)) into callBackName
--put "mUnknown:" && which && "(" & ¬
--  callBackName & ")"
--
--
```

Exhibit 3

301

NETSCREEN

302 → NetScreen [ ] port

303 → Server 13

304 → Restricted to 13

305 → 8 11 AM

# Exhibit 4

```
1  property justLoaded : false
2  property folderPath : ""

3  on run
4      dolt()
5      set justLoaded to true
6  end run

7  on idle
8      set mins to (time of (current date)) / minutes
9      set triggerMin to 0
10     if mins > triggerMin + 30 then
11         if justLoaded then
12             set justLoaded to false
13         end if
14     else
15         if not justLoaded and mins ≥ triggerMin then
16             dolt()
17             set justLoaded to true
18         end if
19     end if
20 end idle

21 on dolt()
22     if folderPath = "" then
23         set folderPath to ((path to (the preferences folder)) as string) & "WebTrio Documents:"
24     end if
25     doDesktopDisplay()
26     doNetscapeDisplay()
27     doScreenSaverDisplay()
28 end dolt

29 on doDesktopDisplay()
30     set wasDeskPictureRunning to isProcessRunning("CLY7")
31     if wasDeskPictureRunning then
32         tell application "DeskPicture" to quit
33     end if
34     set wasFrontierRunning to isProcessRunning("LAND")
35     set fileList to (list folder of (folderPath & "For the Desktop:"))
36     set urlList to {}
37     repeat with fileName in fileList
38         set urlList to getFileComment(alias (folderPath & "For the Desktop:" & fileName)) & urlList
39     end repeat
40     if not wasFrontierRunning then
```

41 tell application "Frontier" to quit  
 42 end if

43 set fileList to (list folder (folderPath & "For the Desktop:"))  
 44 fetchAllPictures(urlList, folderPath & "For the Desktop:", fileList)  
 45 convertToPictAllPicturesIn(folderPath & "For the Desktop:", fileList)

46 if wasDeskPictureRunning then  
 47 tell application "DeskPicture" to run  
 48 end if  
 49 end doDesktopDisplay

50 on doNetscapeDisplay()  
 51 set wasFrontierRunning to isProcessRunning("LAND")  
 52 set fileList to (list folder of (folderPath & "For Netscape"))  
 53 set urlList to {}  
 54 repeat with fileName in fileList  
 55 set urlList to urlList & getFileComment(alias (folderPath & "For Netscape:" & fileName))  
 56 end repeat

57 if not wasFrontierRunning then  
 58 tell application "Frontier" to quit  
 59 end if

60 tell application "Netscape Navigator™ 3.0"  
 61 make new document  
 62 end tell

63 fetchAllPictures(folderPath & "For Netscape:", "", false)  
 64 end doNetscapeDisplay

65 on doScreenSaverDisplay()  
 66 end doScreenSaverDisplay

67 on fetchAllPictures(urlList, folderPath, fileList)  
 68 set wasNetscapeRunning to isProcessRunning("MOSS")  
 69 tell application "Netscape Navigator™ 3.0"  
 70 set netscapeAlertApp to the alert application  
 71 set alert application to "zzzz" -- Asks Netscape not to display alert boxes  
 72 end tell

73 repeat with i from 1 to (length of urlList)  
 74 set myURL to item i of urlList  
 75 if folderPath ≠ "" then  
 76 set fileName to item i of fileList  
 77 netscapeGetURL(myURL, (folderPath & fileName & ".1"), 5, 300)  
 78 else  
 79 netscapeGetURL(myURL, "", 5, 300)  
 80 end if  
 81 end repeat

82 if wasNetscapeRunning then



```

--83 tell application "Netscape Navigator™ 3.0"
84     set alert application to netscapeAlertApp -- Resume Netscape alert boxes display handling
85 end tell
86 else
87     tell application "Netscape Navigator™ 3.0" to quit
88 end if
89 end fetchAllPictures

```

```

90 on convertToPictAllPicturesIn(folderPath, fileList)
91     set wasClip2GifRunning to isProcessRunning("c2gf")
92     set wasJPegViewRunning to isProcessRunning("JVWR")

93     repeat with fileName in fileList
94         set fileAlias to (alias (folderPath & fileName))
95         convertToPict(folderPath & fileName & "1", fileName & "1")
96         «event ScTIExch» (alias (folderPath & fileName & "1")) given «class with»:(fileAlias)
97         «event ScTidele» (alias (folderPath & fileName & "1"))
98     end repeat

99     if not wasClip2GifRunning then
100         tell application "clip2gif" to quit
101     end if
102     if not wasJPegViewRunning then
103         tell application "JPegView" to quit
104     end if
105 end convertToPictAllPicturesIn

```

-----NETSCAPE RELATED ROUTINES-----

```

106 on netscapeGetURL(myLoc, destFile, nbOfTries, myTimeOut)
107     set errCounter to 0
108     repeat while errCounter < nbOfTries
109         tell application "Netscape Navigator™ 3.0"
110             with timeout of myTimeOut seconds
111                 repeat while the busy of window 1 ≠ 0
112                     end repeat
113                 set isLoading to true
114                 GetURL myLoc to (file destFile)
115                 set isLoading to false
116                 repeat while not isLoading
117                     try
118                         the busy of window 1
119                         set isLoading to true
120                     on error
121                     end try
122                 end repeat
123             end timeout
124         try
125             if the file type of (info for (file destFile)) = "TEXT" then
126                 set errCounter to errCounter + 1
127                 «event ScTidele» destFile
128             else
129                 return false -- no error

```

```

130
131         end if
132         on error
133             set errCounter to errCounter + 1
134         end try
135         end tell
136     end repeat
137     return true -- error
138 end netscapeGetURL

```

```

138 on «event WWW?PRBG»
139     return 1
140 end «event WWW?PRBG»

```

```

141 on «event WWW?PRMK»
142     return 0
143 end «event WWW?PRMK»

```

```

144 on «event WWW?PREN»
145     set finished to true
146     return 0
147 end «event WWW?PREN»

```

-----FINDER RELATED ROUTINES-----

```

148 on isProcessRunning(procString)
149     repeat with processName in (list processes)
150         if signature of (get process processName) = procString then
151             return true
152         end if
153     end repeat
154     return false
155 end isProcessRunning

```

```

156 on getFileComment(fileAlias)
157     tell application "Frontier"
158         lfile.getComment1(fileAlias)
159     end tell
160 end getFileComment

```

-----PICTURE CONVERSION ROUTINE-----

```

161 on convertToPict(filePath, fileName)
162     try --We check whether the file exists
163         set fileType to the file type of (info for (file filePath))
164     on error
165         return
166     end try
167     if fileType = "JPEG" then
168         tell application "JpegView"
169             try

```

```
170         open {alias filePath}
171         save document 1 in (alias filePath) as picture
172         close document 1
173     on error
174     end try
175 end tell
176 else
177     if fileType = "GIF" then
178         tell application "clip2gif"
179             try
180                 open (file filePath) given «class fltp»:picture, «class kfil»:(file (filePath & "2"))
181                 «event ScTldele» (alias filePath)
182                 «event ScTIRena» (alias (filePath & "2")) given «class name»:fileName
183             on error
184             end try
185         end tell
186     end if
187 end if
188 end convertToPict
```